# THOMSON - BRANDT

DPT : ORDINATEUR FAMILIAL

N° : 1-4-45

Rev : 1

CLASSIFICATION : DOCUMENT DE TRAVAIL

DATE : 28/04/86

TACHE : MACHINE 86

SOUS-TACHE : Logiciel

RELECTURE :

AUTEUR : HH

TITRE : PRESENTATION OF GRAPHIS MODULE

MOTS CLES : Graphport, VSDD, Bitmap, Brush, Pattern

REFERENCES :

RESUME :

This document describes the graphic entities GRAPHIS deals with, and gives a quick summoning of its routines and their possibilities.

For more informations about the entry points, please see current version of document 1-4-47.

REVISION : 1

FAITE PAR : HH

LE : 24/10/85

DESTINATAIRES :

ZW, JRM, PP, ER, HH, TS, PaP, BR, PW + G. Corda, H. Bürklin, Mr. Zeitlin.

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

# PRESENTATION OF GRAPHIS MODULE

The goal of GRAPHIS module is to handle graphics drawings in a bitmap of given size and color resolution (1, 2 or 4 bits per pixel), in a given memory area. This module doesn't "know" anything about the VSDD, which is handled by the VIDEOMAN module.

## 1 GRAPHIC ENTITIES

### 1.1 THE BITMAP

The bitmap corresponds exactly to the VSDD object, whose width must be a multiple of 64 bits in the current version of the chip.

As said above, the VSDD programming is done by the VIDEOMAN module; we'll only consider here the bitmap organization:

To figure out the screen-memory organization, we number pixels from left to right and from top to bottom, each pixel corresponding to a group of 1, 2 or 4 bits; these bits are then grouped together into words, and these words are inverted.

Here are a few examples in the different color resolutions:

#### 1.1.1 One Bit Per Pixel

WORD 0: pixels 0 through 15

| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

WORD 2: pixels 16 through 31

| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

and so on ...

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

### 1.1.2 Two Bits Per Pixel

WORD 0: pixels 0 through 7

| b15 b14 | b13 b12 | b11 b10 | b9 b8 | b7 b6 | b5 b4 | b3 b2 | b1 b0 |
|---|---|---|---|---|---|---|---|
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

WORD 2: pixels 8 through 15

| b15 b14 | b13 b12 | b11 b10 | b9 b8 | b7 b6 | b5 b4 | b3 b2 | b1 b0 |
|---|---|---|---|---|---|---|---|
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |

and so on ...

### 1.1.3 Four Bits Per Pixel

WORD 0: pixels 0 through 3

| b15 b14 | b13 b12 | b11 b10 | b9 b8 | b7 b6 | b5 b4 | b3 b2 | b1 b0 |
|---|---|---|---|---|---|---|---|
| P3 | | P2 | | P1 | | P0 | |

WORD 2: pixels 4 through 7

| b15 b14 | b13 b12 | b11 b10 | b9 b8 | b7 b6 | b5 b4 | b3 b2 | b1 b0 |
|---|---|---|---|---|---|---|---|
| P7 | | P6 | | P5 | | P4 | |

and so on ...

## 1.2 THE "LOGICAL OBJECT"

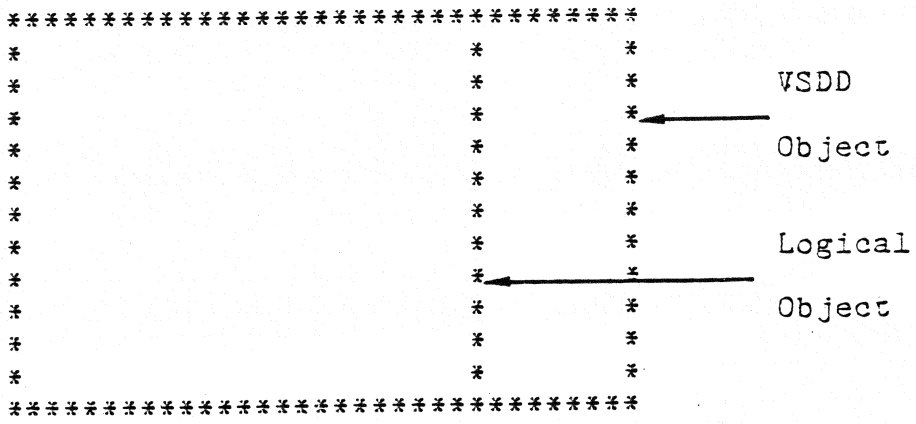
The logical object has the same color resolution and height than the VSDD object, but may have a width lower than or equal to the VSDD object's.

The distinction between the VSDD object and the "logical" object is made to bypass the constraint on the VSDD width, which, as said before, must be a multiple of 64 bits, that is to say 64, 32 or 16 pixels, depending on the color resolution.

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

If an application program wants to create an object whose width is not a multiple of 64 bits, then pixels between logical object and VSDD bitmap have to be transparent: this of course excludes the 1 bit/pel resolution, for there would be only one color left (note that although GRAPHIS handles the 1 bit/pixel resolution, only 2 and 4 bits/pixel resolutions are provided by the current version of the chip).

Though the application program has to know (for proper color handling for example) the existence of a transparent area, it is allowed to write only into the logical object it defined.




In this paper, we'll use the word "object" to mean "logical object" and "VSDD object" or "bitmap" for the actual VSDD object.

All information about location, placement, or movement given to GRAPHIS is in terms of coordinates on a plane. Horizontal coordinates ranges from -32768 to +32767, and increase as you move from left to right. Vertical coordinates have the same range and increase as you move from top to bottom. The origin coincides with the object's upper left corner.

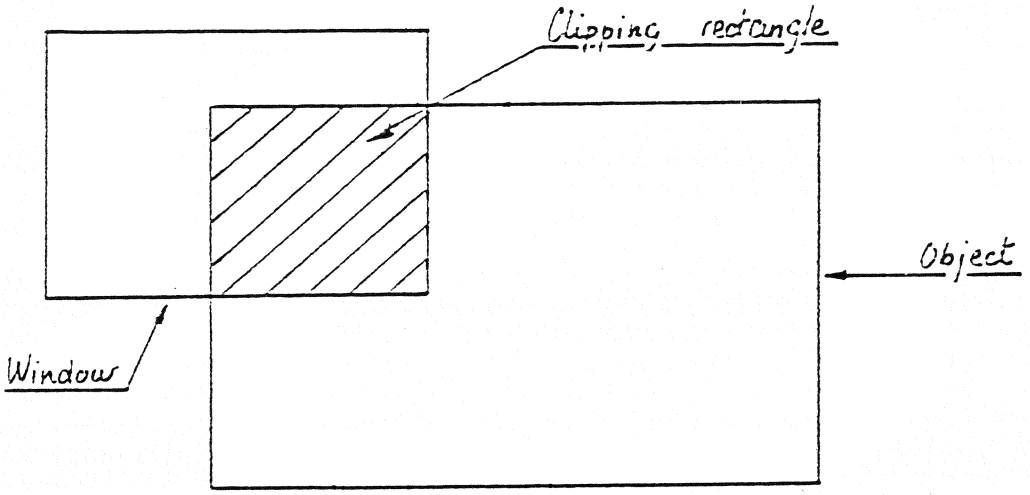
The object has an object mode which determines the routines' behaviour when any of the input coordinates are out of the object's range: if the object is in clipping mode, it will not generate an error; otherwise, if the object is in error mode, the appropriate error code will be returned.

## 1.3 REGIONS AND WINDOWS

An application program may define a window or a region. Two points define the top left and right bottom corners of a window, whose intersection with the object must at least enclose one pixel. A region is a set of disjoint windows, stored in memory as a collection of coordinates. The number of windows of a region ranges from 2 to 256.

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

Note that whatever's being drawn is clipped to the rectangle(s) that is (are) the intersection of the object and the window(s), so that it doesn't run over outside the object.




Similar to the object mode, there's a window mode: it determines the behaviour of the drawing routines when part of the drawing is outside the clipping area: clipping or appropriate error code returning. Note that this mode is forced to clipping mode when a region is currently defined, for its windows are disjoint.

## 1.4 DRAWING CHARACTERISTICS

Drawing routines are divided into two sets: pen routines, which draw basic geometrical shapes, and brusn routines which can either draw an hollow outline inside the specified shape, or fill it, according to the current brush and pattern.

### 1.4.1 The Current Location

The current location is a point in the coordinate system of the object, and is refered to as a start point in many drawings. There is no other restriction on the placement of the current location than the object mode.

### 1.4.2 The Current Colors

The background color, is used for object and window clearing, pattern drawings, and the pen color is used for pen (1 pixel thick) and pattern drawings.

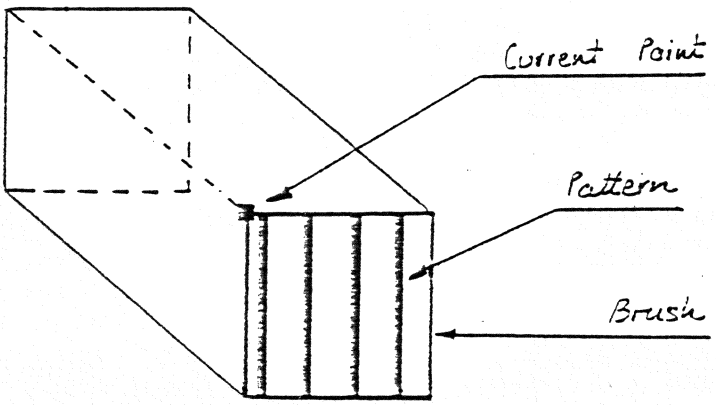
Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

### 1.4.3 The Brush

The brush is an 8 by 8 or 16 by 16 bits square, stored in memory as a collection of 64 or 256 bits, each corresponding to a pixel in the brush and specifying which points will be displayed (bits set) and which won't (bits clear), that is, define the brush shape.

The brush also has an origin that specifies the location of the brush relative to the current point. Its ranges from (0,0) to (7,7) for an 8 by 8 bits brush, and from (0,0) to (15,15) for a 16 by 16 bits brush.

For instance, if the brush is defined by 64 bits set, and its origin is (0,0), it will appear as a square with its top left corner at the current point, hanging below and to the right of it.




### 1.4.4 Patterns

A pattern is an 8 by 8 or 16 by 16 bits square, which is used to define a repeating color design. Patterns can be used to draw shapes or to fill them, on a "virtual pattern grid". As with the brush, each bit corresponds to a pixel. Bits set correspond to pen color, while bits clear correspond to background color, unless the pattern function is not the copy one: the pattern function can also be invert copy (exchange background and pen colors), 0's transparency (ignore pattern's bits clear and background color), or 1's transparency (ignore pattern's bits set and pen color).

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

## 2 THE GRAPHPORT

A Graphport is a data structure created by VIDEOMAN, which can be divided into three main zones:

### 2.1 PERMANENT AND NON-ALTERABLE ZONE

This first zone is initialized by VIDEOMAN when the VSDD object is created (or moved...), and is only read by GRAPHIS. Fields in this zone deal with parameters of the concerned VSDD object, in a suitable form for graphics routines. It is defined as follows:

* GPBMHDL: 32 bits, VIDEOMAN register.
* GPBMWID: 16 bits, VIDEOMAN register.
* GPBMDEP: 16 bits, VIDEOMAN register.
* GPTYP: 8 bits, VIDEOMAN register.
* GPRES: 8 bits, VIDEOMAN register.
* GPBMSIZ: 16 bits, size in words of the VSDD object.
* NBBYTE: 16 bits, number of bytes per line of the VSDD object.
* PIXMAX: 16 bits, number of the logical object's rightmost column.
* LINMAX: 16 bits, number of the logical object's bottom line.
* DIFOBJ: 8 bits, width (in pixels) of the transparent area between object and VSDD object (ranges from 0 to 63, 31 or 15, depending on the color resolution).
* COLRES: 8 bits, color resolution (2 ou 4 bits/pel).
* LOGRES: 8 bits, Log2 (COLRES).
* NPIXWD: 8 bits, number of pixels per word.

### 2.2 PERMANENT AND ALTERABLE ZONE

Fields in this zone deal with the drawing environment, which can be modified by GRAPHIS under an application

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

program's request. They're initialized to default values by VIDEOMAN. The zone is defined as follows:

2.2.1 Region, Window, Current Location and Colors Fields

* LOCATY: 16 bits, current location's Y-coord.
* LOCATX: 16 bits, current location's X-coord.
* BCKCOL: 8 bits, background color.
* PENCOL: 8 bits, pen color.
* CLPTOP: 16 bits, window's top line, or signed displacement to add to the top line of each window of a region.
* CLPBOT: 16 bits, window's bottom line, or signed displacement to add to the bottom line of each window of a region.
* CLPLFT: 16 bits, window's left column, or signed displacement to add to the left column of each window of a region.
* CLPRGT: 16 bits, window's right column, or signed displacement to add to the right column of each window of a region.
* RGNPTR: 32 bits, region definition pointer (clear if a window is currently defined).
* OBJMOD: 8 bits, object mode (error or clipping).
* CLPMOD: 8 bits, window mode (error or clipping if a window is currently defined, otherwise clipping).
* RGNSIZ: 8 bits, 0 if a window is currently defined, n-1 if a region of n windows is currently defined.
* BNDFLG: 8 bits, 0 if the window(s) is (are) enclosed in the object.

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

### 2.2.2 Brush And Pattern Fields

* PATPTR: 32 bits, pattern pointer.
* PATSIZ: 8 bits, pattern size - 1 (7 or 15).
* PATFUN: 8 bits, pattern function (1, -1, 2 or -2).
* PATSTP: 8 bits, for brush-routines internal use.
* PATMAX: 8 bits, for brush-routines internal use.
* BRSHAP: 32 bits, brush shape pointer.
* BRUSHY: 8 bits, brush origin's Y-coord.
* BRUSHX: 8 bits, brush origin's X-coord.
* BSHSIZ: 16 bits, brush size - 1 (7 or 15).
* BSHSTP: 16 bits, for brush-routines internal use.
* BSHMAX: 8 bits, for brush-routines internal use.

### 2.2.3 Font Fields

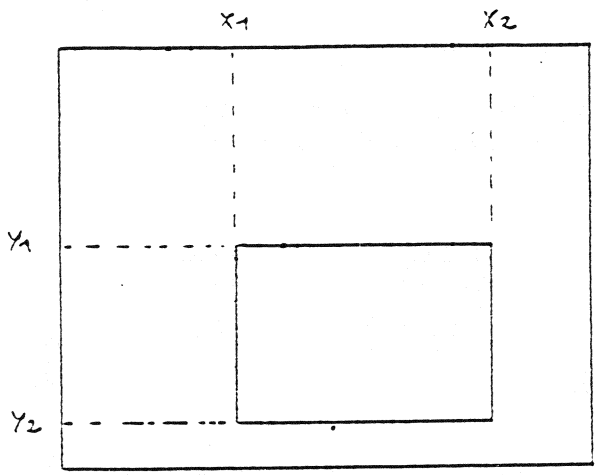
* GENPTR: 32 bits, character generator pointer.
* BLDFAC: 8 bits, bold factor.
* ITLFAC: 8 bits, italic factor.
* UNOFFT: 8 bits, offset in pixels for underline display.
* UNTHCK: 8 bits, underline thickness.
* UNSHDW: 8 bits, underline shadow.
* EXTRWD: 8 bits, extra width value caused by style.
* SCLFLG: 8 bits, scaling flag.
* EXTRSP: 8 bits, extra space.
* CHARC0: 16 bits, for font-routines internal use.
* CHARC1: 16 bits, for font-routines internal use.
* ASCENT: 16 bits, number of ascent lines.
* DSCENT: 16 bits, number of descent lines.

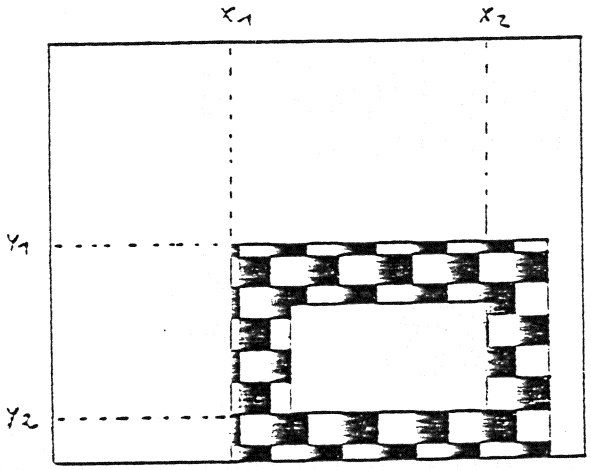
Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

* WIDMAX: 16 bits, character width in pixels.
* RCWDMX: 16 bits, rectangle width in pixels.
* UPLNES: 16 bits, number of blank lines at the character top.
* VRTNOM: 16 bits, numerator of the vertical scale factor.
* HRZNOM: 16 bits, numerator of the horizontal scale factor.
* VRTDNM: 16 bits, denominator of the vertical scale factor.
* HRZDNM: 16 bits, denominator of the horizontal scale factor.
* STYLEF: 16 bits, for font-routines internal use.
* EXTRZN: 16 bits, for font-routines internal use.
* EXTNZN: 16 bits, for font-routines internal use.
* TXTMOD: 8 bits, text mode.

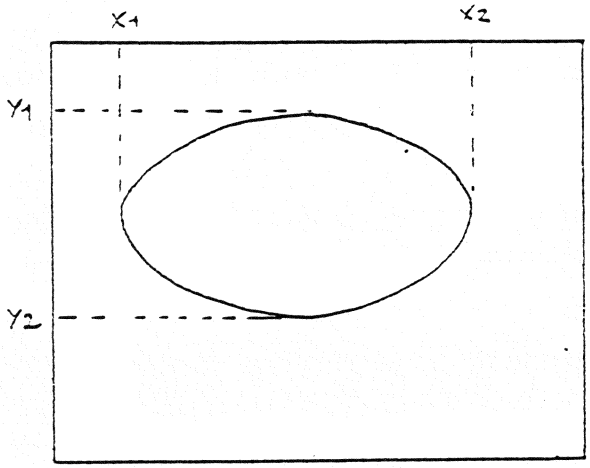
Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

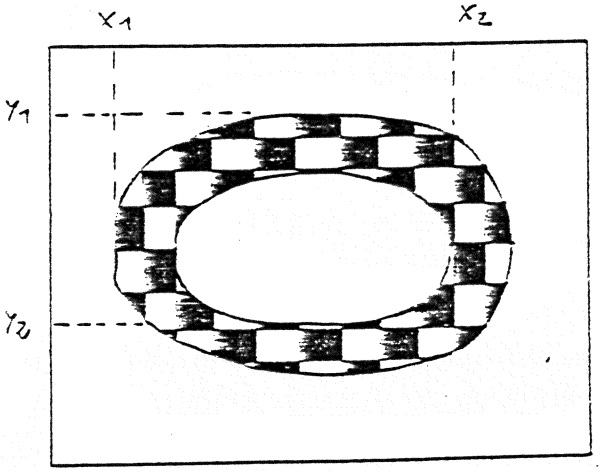
### 3.1.4 Rectangles



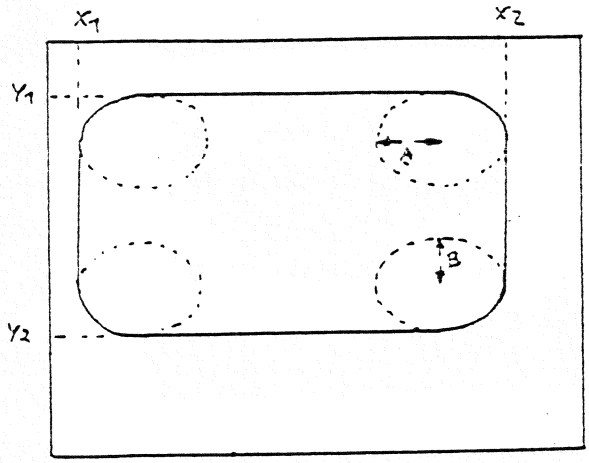



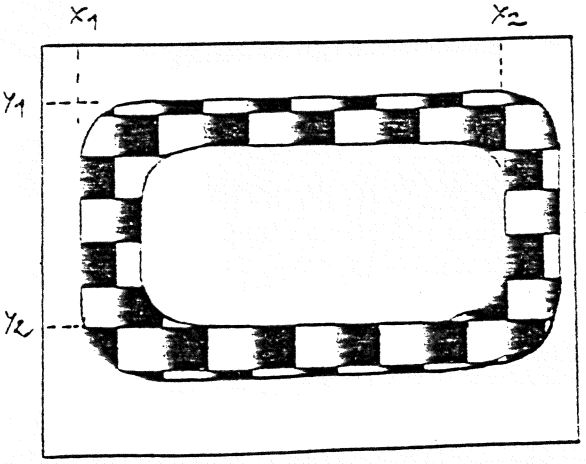

### 3.1.5 Ellipses



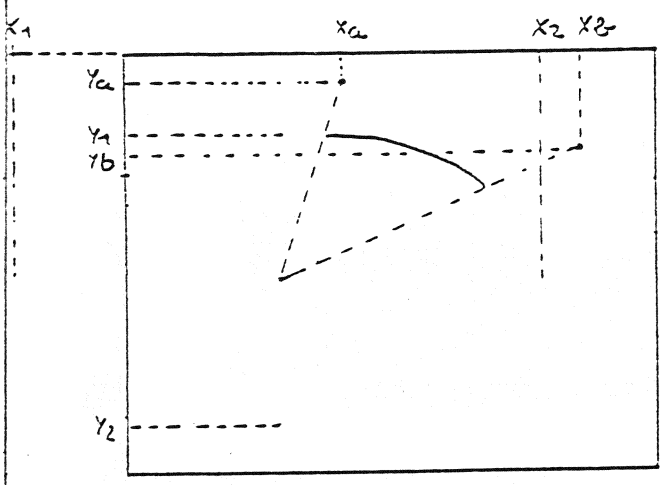

### 3.1.6 Oval-rectangles

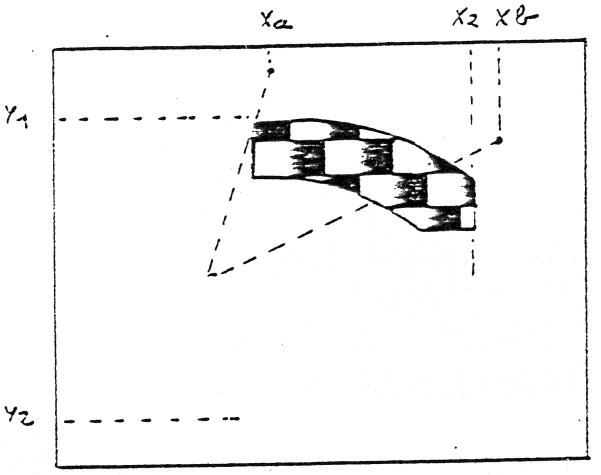


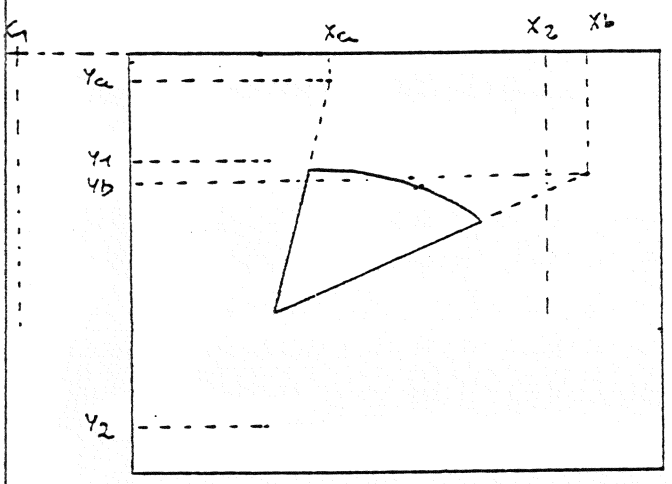

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

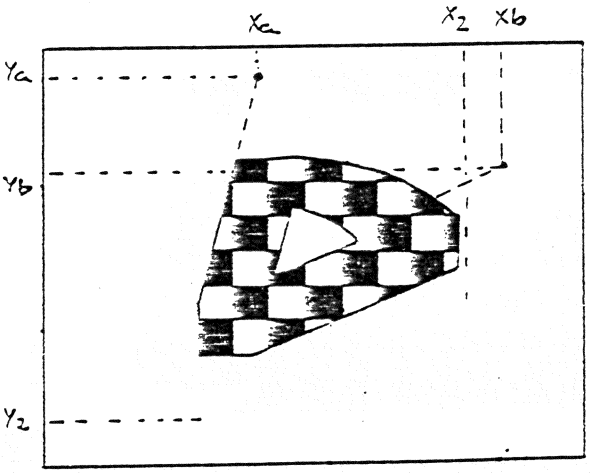
### 3.1.7 Arcs



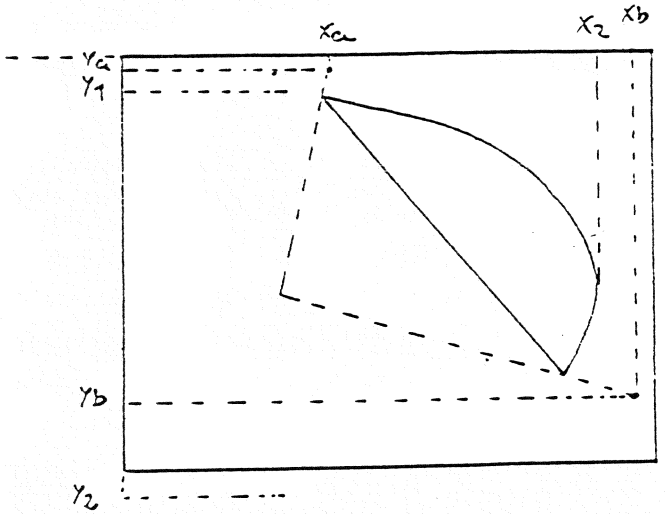



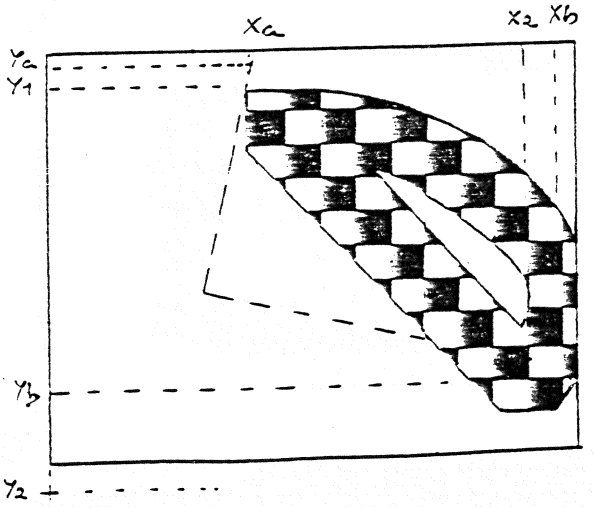

### 3.1.8 Pie Slices




### 3.1.9 Bows




Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

## 3.2 ENTRY POINTS

Graphics routines are called using TRAP instruction, according to OS-9/68000 conventions. Here follows a quick summoning. For more details, please see Thomson's document 1-4-47.

### 3.2.1 Get And Set Graphics Environment

11 routines to get or set background, pen, object and window modes, region or window definition, current location, brush size, pointer and origin, pattern size, pointer and function, or to get the color of a given pixel.

* SETCOL modifies pen and/or background color.

* SETBSH changes brush size, origin, or shape.

* SETPAT changes pattern size, function, or definition.

* MOVLOC, SETLOC change the current location.

* GETSTA returns pen and background colors, the current location, the brush size, origin and pointer, the pattern size, function and pointer.

* SETMOD changes the object or window mode.

* MOVRGN, SETRGN change the region or window.

* GETRGN returns object and window modes, and either the coordinates of window's corners, or the region pointer, number of windows, and signed displacements that are to be added to the top line, bottom line, left column and right column of each window of the region.

* GETCOL returns the color of a given pixel.

### 3.2.2 Bitmap Handling

5 routines to clear or fill the object, fill or scroll window(s).

* CLBMAP clears the whole VSDD object.

* CLROBJ fills the logical object with a given color.

* CLRWDW fills the window with a given color.

* VLSCRL scrolls down or upwards the window(s) contents of a given number of lines, blank entering lines being filled with a given color.

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

* HLSCRL scrolls right or leftwards the window(s) contents of a given number of columns, blank entering lines being filled with a given color.

### 3.2.3 Pen Routines

27 routines to draw pixels, lines, polylines, rectangles, ellipses, rectangles with elliptical corners, elliptical arcs, pie slices or "bows".

* PIXELA, PIXELR, PIXELC draw a dot in a given color.

* LINEAB, LINERE, LINECL draw a line in a given color.

* POLYLA, POLYLR, POLYLC, draw a polyline in a given color.

* RCTGLA, RCTGLR, RCTGLC draw a rectangle in a given color.

* ELLPSA, ELLPSR, ELLPSC draw an ellipse in a given color.

* OVRECA, OVRECR, OVRECL draw a rectangle with elliptical corners, in a given color.

* ARCELA, ARCELR, ARCELC draw an elliptical arc in a given color.

* PIECTA, PIECTR, PIECTC draw an elliptical pie slice in a given color.

* ELBOWA, ELBOWR, ELBOWC draw an elliptical bow in a given color.

### 3.2.4 Brush Routines

27 routines to draw, according to the current brush and pattern, pixels, lines, polylines, rectangles, ellipses, rectangles with elliptical corners, elliptical arcs, pie slices or bows.

* PIXPNA, PIXPNR, PIXPNC paint a dot using current brush and pattern, in given colors.

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

* LINPNA, LINPNR, LINPNC paint a line using current brush and pattern, in given colors.

* POLPNA, POLPNR, POLYNC, paint a polyline using current brush and pattern, in given colors.

* RECPNA, RECPNR, RECPNC paint a rectangle using current brush and pattern, in given colors.

* ELLPNA, ELLPNR, ELLPNC paint an ellipse using current brush and pattern, in given colors.

* OVRPNA, OVRPNR, OVRPNC paint a rectangle with elliptical corners, using current brush and pattern, in given colors.

* ARCPNA, ARCPNR, ARCPNC paint an elliptical arc using current brush and pattern, in given colors.

* PIEPNA, PIEPNR, PIEPNC paint an elliptical pie slice, using current brush and pattern, in given colors.

* BOWPNA, BOWPNR, BOWPNC paint an elliptical bow, using current brush and pattern, in given colors.

## 3.3 ERROR HANDLING

GRAPHIS routines handle errors according to OS-9/68000 conventions, that is, set carry and return error code in register D1.W if an error occured, otherwise clear carry and D1.W.

Errors are:

* Overflow: this error can occur, for instance, if a 16-bit signed displacement has to be added to a 16-bit coordinate.

* Out of object's range: this error occurs if a coordinate is out of the logical object's range, and the object is in error mode.

* Out of window: this error occurs if the drawing is out of the rectangle that is the intersection of the current window and the object; this error can't occur if a region is currently defined.

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse

* Bad color: this error occurs if the specified color code is not compatible with the color resolution of the object.

* Bad parameter: this error occurs if an input parameter in not compatible with the routine's specifications, or if two parameters are not compatible (for instance, if the right column of a window is lower than the left one).

Propriété de THOMSON - BRANDT, ce document ne peut-être communiqué aux tiers sans autorisation expresse